Optimum Equipment Maintenance/Replacement Policy Part 2. Markov Decision Approach

T. Charng
DSN Engineering Section

This is the second article on the subject of Optimum Equipment Maintenance and/or Replacement Policy which employs the optimization technique called Markov Decision Process. In the first article, dynamic programming was utilized as an alternative optimization technique to determine an optimal policy over a given time period. According to a joint effect of the probabilistic transition of states and the sequence of decision making, the optimal policy is sought such that a set of decisions optimizes the long-run expected average cost (or profit) per unit time. Provision of an alternative measure for the expected long-run total discounted cost is also considered. A computer program based on the concept of the Markov Decision Process was developed and tested. The program code listing, the statement of a sample problem, and the computed results are presented in this report.

I. Introduction

In the first article (*TDA Progress Report 42-66: September and October 1981*), an optimal decision-making policy utilizing the dynamic programming technique was presented. The aim was to make optimal decisions, over a finite number of time periods, regarding the equipment maintenance and/or replacement for a given system.

When a system is required to be existing indefinitely, a best operation policy which gives the optimal long-run cost (or profit) may be estimated by successive approximations with dynamic programming techniques, providing a very large number of time periods is assumed. However, there is no way of knowing when to terminate the successive approximations; there is no procedure for deciding how large a number of time periods is sufficient.

This report presents another decision-making technique to obtain a long-run optimal policy utilizing the Markov Decision Process Concept. The optimal policy is evolved over time periods according to the joint effect of the probabilistic transition of the condition of the system and the sequence of decision making. It is assumed that for a system under consideration, there exists a policy at any time period. The system changes to a new state at the next time period according to a known probability after a decision is made at the present time period and at the present state.

In such decision making, different transition matrixes result, corresponding to the decisions made at each observed time period. Accordingly, a change of state as well as the associated value of cost (or profit) of the system in the next time period is governed by the transition matrix. The optimal is sought such that the set of decisions made will optimize the long-run expected average cost (or profit) per unit time period.

However, in systems involved with a long time-horizon, the changing time value of money is of importance, and the expected long-run total discounted cost (or profit) should be determined with respect to a specific discount factor. The discount factor a is the present value of one unit money in one time period, expressed as:

$$a = \frac{1}{1+i} \tag{1}$$

where i is the rate of return on the money for one time period. After M periods, a unit of money will be worth a^M.

Derivation of mathematical equations, discussion of the policy-improvement computational algorithm, and a sample problem are presented in the following sections. A computer program employing the discussed algorithm is given in Appendix A.

II. Theoretical Model

Brief descriptions of the Markov Decision Process and the necessary equations are discussed as follows. More detailed derivations of the equations may be found in Refs. 1, 2.

Consider a system which at a particular time period ($t = 1, 2, 3, \ldots$) is in one state i out of M states. The system changes from one of these admissible states, i, to another state, j, ruled by a transition matrix, $P = (p_{ij})$. The elements, p_{ij}, are defined as the probabilities the system is in state j at t, given that it was in state i at $(t-1)$. Further, it is assumed that the transition matrix P is not time-dependent.

Let $q_{ij}(k)$ be the expected cost (or profit) incurred when the system, which originally is in state i, changed after a decision k is made to a state j at the next observed time period. Then,

$$C_{ik} = \sum_{j=1}^{M} q_{ij}(k)\, p_{ij}(k) \tag{2}$$

where C_{ik} is the cost incurred at the first observed time period as a result of the current state i and the decision $D_i(R) = k$ when operating under policy R.

By introducing $V_i^N(R)$ as the total expected cost of a system starting in state i (at the first observed time period) and evolving for N time periods following a policy R, the recursive equation can be written as

$$V_i^N(R) = C_{ik} + \sum_{j=1}^{M} p_{ij}(k)\, V_j^{(N-1)}(R) \tag{3}$$

The second term on the right-hand side of Eq. (3) is the total expected cost of the system evolving over the remaining $(N-1)$ time periods.

Let $g(R)$ be the long-run expected average cost per unit time following a policy R. As one of the Markovian properties, it can be shown that the $g(R)$ is independent of the starting state i as the number of time periods N approaches infinity. Hence, $V_i^N(R)$ may be approximated by

$$V_i^N(R) \cong N\, g(R) + V_i(R) \tag{4}$$

where $V_i(R)$ can be interpreted as the effect on the total expected cost due to starting in state i. Thus, from Eq. (4)

$$V_i^N(R) - V_j^N(R) \cong V_i(R) - V_j(R) \tag{5}$$

the term $[V_i(R) - V_j(R)]$ is a measure of the effect of starting in state i rather than state j. Substituting the linear, approximate relations of Eqs. (4, 5) into Eq. (3) leads to the recursive equation

$$g(R) + V_i(R) = C_{ik} + \sum_{j=1}^{M} p_{ij}(k)\, V_j(R) \tag{6}$$

Equation (6) represents one of the M equations corresponding to the state i for $i = 1, 2, \ldots, M$.

When a system operates according to the Markov chain, there are needed $(M+1)$ values of $g(R)$, $V_1(R)$, $V_2(R), \ldots, V_M(R)$ which satisfy the set of M equations of the form Eq. (6). Note that there are M equations and $(M+1)$ unknowns; one of the unknowns, say $V_M(R)$, can be arbitrarily set to equal zero. Following a given policy R, the corresponding values of $g(R)$, $V_1(R)$, $V_2(R), \ldots, V_{(M-1)}(R)$ can then be obtained by solving the set of M simultaneous linear equations.

In principle, all policies can be enumerated to find the policy which optimizes the $g(R)$. However, even for a moderate number of states and decisions, this enumeration technique is cumbersome. A different approach called Policy-Improvement can be used to evaluate policies and find the optimal set of decisions without a complete enumeration. The mechanism of this algorithm is presented in the next section.

III. Computational Algorithm

The Policy-Improvement algorithm consists of two steps: the Value-Determination step and the Policy-Improvement step. These steps are described as follows.

(1) Value-Determination Step: For an arbitrary policy R_1 (with decisions $D_i(R_1) = k_1$, and the corresponding values of $p_{ij}(k_1)$, C_{ik_1}, and $V_M(R_1) = 0$), this step solves the set of M equations of Eq. (6), or

$$g(R_1) + V_i(R_1) = C_{ik} + \sum_{j=1}^{M} p_{ij}(k_1)\, V_j(R_1) \tag{7}$$

for $i = 1, 2, \ldots, M$. Hence, values of the $g(R_1)$, $V_1(R_1)$, $V_2(R_1), \ldots, V_{(M-1)}(R_1)$ are obtained under policy R_1.

(2) Policy-Improvement Step: Using the above calculated values of the V's, find the alternative policy R_2 such that for each state i, $D_i(R_2) = k_2$ is the decision which optimizes $g(R_2)$, with

$$g(R_2) = C_{ik} + \sum_{j=1}^{M} p_{ij}(k_2)\, V_j(R_1) - V_i(R_1) \tag{8}$$

That is, for each state $i = 1, 2, \ldots, M$, find the appropriate value of k_2 such that

$$\underset{k_2 = 1, 2, \ldots, K}{\text{OPTIMUM}} \quad [g(R_2)] \tag{9}$$

In turn, let $D_i(R_2)$ be equal to the optimal value of k_2, which defines a new policy R_2.

Using the new policy R_2, the Value-Determination step is repeated. This iterative procedure continues until two successive iterations lead to identical policies, which signifies that the optimal policy has been obtained.

If the expected long-run total discounted cost is of interest, the above algorithm can be used with a modification to the recursive equation Eq. (6). With a specified discount factor a, the recursive relation of Eq. (6) can be modified as

$$V_i(R) = C_{ik} + a \sum_{j=1}^{M} p_{ij}(k)\, V_j(R) \tag{10}$$

for $i = 1, 2, \ldots, M$. Here, the $V_i(R)$ is the expected long-run total discounted cost of the system starting in state i and continuing indefinitely. The $V_i(R)$ can be evaluated in a similar fashion as computing the average cost.

A computer program is written to incorporate both the average cost of Eq. (6) and the discount cost of Eq. (10). The optimal policy is determined utilizing the Policy-Improvement algorithm. This BASIC program code is presented in Appendix A. A sample problem, adapted from Ref. 2 and presented in the next section, is used for testing purposes. The calculated results of both the averaged cost and the discounted cost are presented in Figs. 1 and 2, respectively.

IV. Sample Problem

For the purpose of testing the computer program, a sample problem is taken from Ref. 2 and summarized as follows.

The condition of a given system is inspected and classified into one of four possible states as shown in Table 1. It is also assumed that the state of the system evolves according to some known probabilistic transition matrix given in Table 2. After each periodic inspection of the system, a decision must be made as to which action to take: Decision 1 is doing nothing; Decision 2 is overhauling the system; Decision 3 is replacing the system.

In addition, the following assumptions are made:

(1) When the system becomes inoperable (State 4) and replaced (Decision 3), the system is found to be in State 1 at the time of regular inspection. It is assumed that the total cost incurred when the system is in State 4 is the sum of a replacement cost of \$4000 and a cost of lost production of \$2000.

(2) When the system is overhauled, the system is returned to State 2 (operable with minor deterioration) at the time of regular inspection at the end of next time period. The cost of the overhaul process is taken as \$2000 and requires one time period to complete.

(3) When the system is in States 2 or 3, defective items may be produced during the following operating period. The expected costs due to producing defective items are \$1000 when the system is operable with minor deterioration and \$3000 when the system is operable with major deterioration.

(4) The total expected cost incurred per one time period depends on the state the system is in and the decision made. The total expected costs (the maintenance cost, the cost due to producing defective items, and the cost from lost production) are tabulated in Table 3.

The above information completed the necessary inputs to the computer program. Figure 1 presents the optimal policies and the expected average cost of the sample problem. As the result of the Markov Decision Process, an average cost of \$1667 can be expected when the policy is to do nothing when the system is found to be in States 1 and 2, to overhauling the system when it is in State 3, and to replace the system when it is in State 4.

In the second case, as presented in Fig. 2, an interest rate of 11% (or discount factor of 0.9) was assigned. However, with the same policy as in Case 1, a discounted cost of \$14,950 can be expected if the system started in State 1, \$16,260 if it started in State 2, etc.

V. Summary

The Policy-Improvement algorithm using a Markov Decision Process is incorporated in a computer program and tested with a sample problem on a Hewlett-Packard 2647A terminal. This computer program is capable of finding the best maintenance policy with respect to an optimal long-run average cost or the long-run discounted cost for a system with known transition probabilities.

From the standpoint of management and operation, the algorithm provides a useful tool in obtaining an optimal maintenance schedule which gives the best return on capital invested.

Acknowledgment

The author thanks Dr. F. L. Lansing, who made a number of helpful suggestions in the study preparation.

References

1. Bellman, R. E., and Dreyfus, S. E., *Applied Dynamic Programming*, Princeton University Press, Princeton, N.J., 1962.

2. Hillies, F. S., and Lieberman, G. J., *Introduction to Operations Research*, 3rd Edition, Holden-Day, Inc., San Francisco, Calif., 1980.

Table 1. States of the system

State	Condition
1	Good as new
2	Operable with minor deterioration
3	Operable with major deterioration
4	Inoperable

Table 2. Transition matrix of the system

State	1	2	3	4
1	0	7/8	1/16	1/16
2	0	3/4	1/8	1/8
3	0	0	1/2	1/2
4	0	0	0	1

Table 3. Total expected cost per one time period

	Decision		
State	1	2	3
1	0	$4000	$6000
2	$1000	$4000	$6000
3	$3000	$4000	$6000
4	–	–	$6000

```
* * * * * INPUT INFORMATION * * * * *


MSTATE       4
NDECISION    3
MAXTRIAL     10
MAXIMUM      0
DISCOUNT     0

STATE        DECISION   TIE-BREAKER
  I            D(I)       TIED(I)
-----        --------   -----------
 1             1          0
 2             1          0
 3             1          0
 4             3          0



* * * * * * *    FINAL RESULTS    * * * * * * *



LONG-TERM AVERAGE COST/RETURN= 1666.67


STATE        POLICY     VALUE
-----        ------     -----
 1             1          1666.67
 2             1          1666.67
 3             2          1666.67
 4             3          1666.67
```

Fig. 1. Expected average cost

```
* * * * * INPUT INFORMATION * * * * *


MSTATE      4
NDECISION   3
MAXTRIAL    10
MAXIMUM     0
DISCOUNT    11.1111

STATE        DECISION   TIE-BREAKER
  I            D(I)       TIED(I)
-----        --------   -----------
  1            1          0
  2            1          0
  3            1          0
  4            3          0


* * * * * * *   FINAL RESULTS   * * * * * * *




STATE        POLICY     VALUE
-----        ------     -----
  1            1          14948.6
  2            1          16261.6
  3            2          18635.5
  4            3          19453.7
```

Fig. 2. Expected discounted cost

Appendix A

Computer Program Listing

```
1 REM.............................................................REM
2 REM.....                                                    .....REM
3 REM.....              EQUIPMENT MAINTENANCE POLICY          .....REM
4 REM.....              MARKOVIAN DECISION ALGORITHM          .....REM
5 REM.....                                                    .....REM
6 REM.............................................................REM
10 DIM Irow(10),Jcol(10),Jord(10),Y(10)
11 DIM Dd(10),D(10),P(5,5,5),R(5,5,5),Sum(5,5),Tied(10),Q(5,5)
15 LONG A(11,11),X(10),Eps,Simul
16 INTEGER D,Dd,Trial,Tied
20 REM
100 REM................ BEGIN OF DATA LIST ........................REM
102 DATA 0,6                   !  ASSIGN I/O DEVICES
105 DATA 0                     !  PRINTOUT OPTION
110 DATA 4                     !  NUMBER OF STATES IN CONSIDERATION
120 DATA 3                     !  NUMBER OF DECISION ALTERNATIVES
130 DATA 0                     !  MAXIMIZE COST/RETURN IF >=1
140 DATA 10                    !   MAXIMUM TRIALS ALLOWED
150 DATA 11.1111              !  DISCOUNT RATE IN WHOLE NUMBER
160 REM................ PROBABILITY MATRIX,  P(I,J,K)  ..............REM
170 DATA 0,.875,.0625,.0625
180 DATA 0,1,0,0
190 DATA 1,0,0,0
200 DATA 0,.75,.125,.125
210 DATA 0,1,0,0
220 DATA 1,0,0,0
230 DATA 0,0,.5,.5
240 DATA 0,1,0,0
250 DATA 1,0,0,0
260 DATA 0,0,0,1
262 DATA 0,1,0,0
264 DATA 1,0,0,0
270 REM................ COST/RETURN MATRIX ( R(I,J,K) )  ............REM
280 REM      DATA 0,0,0,00
290 REM      DATA 0,4000,0,0
300 REM      DATA 6000,0,0,0
310 REM      DATA 0,1000,1000,1000
320 REM      DATA 0,4000,0,0
322 REM      DATA 6000,0,0,0
330 REM      DATA 0,0,3000,3000
340 REM      DATA 0,4000,0,0
350 REM      DATA 6000,0,0,0
360 REM      DATA 0,0,0,1.E30
370 REM      DATA 0,1.E30,0,0
380 REM      DATA 6000,0,0,0
381 REM................ COST/RETURN MATRIX, Q(I,K)=(P(I,J,K)*R(I,J,K).REM
382 DATA 0,4000,6000
383 DATA 1000,4000,6000
384 DATA 3000,4000,6000
385 DATA 1E30,1E30,6000
390 REM................ TIE-BREAKER, TIED(I)  .......................REM
395 DATA 0,0,0,0
400 REM................ INITIAL POLICY, D(I)  .......................REM
410 DATA 1,1,1,3
500 REM................ END OF DATA LIST  ...........................REM
998 REM                                                            REM
```

```
1000 REM...............  GENERAL INPUT  ..............................REM
1005 READ Ki,Ko
1010 REM...............  ASSIGN READ/PRINT FILES  ....................REM
1015 ASSIGN "OUTPUT" TO #Ko
1020 IF Ki<=0 THEN 1190
1025 ASSIGN "INPUT" TO #Ki
1030 READ #Ki;Iprint
1035 READ #Ki;Mstate
1040 READ #Ki;Ndecision
1045 READ #Ki;Maxi
1050 READ #Ki;Maxtrial
1055 READ #Ki;Discount
1060 REM...............  INPUT TRANSITION MATRIX  ....................REM
1065 FOR I=1 TO Mstate
1070 FOR K=1 TO Ndecision
1075 FOR J=1 TO Mstate
1080 READ #Ki;P(I,J,K)
1085 NEXT J
1090 NEXT K
1095 NEXT I
1100 REM...............  INPUT COST/PROFIT MATRIX  ...................REM
1105 FOR I=1 TO Mstate
1110 FOR K=1 TO Ndecision
1115 REM    FOR J=1 TO Mstate
1120 REM    READ #5;R(I,J,K)
1125 REM    NEXT J
1130 READ #Ki;Q(I,K)
1135 NEXT K
1140 NEXT I
1145 REM...............  INPUT TIE-BREAKER FLAG  .....................REM
1150 FOR I=1 TO Mstate
1155 READ #Ki;Tied(I)
1160 NEXT I
1165 REM...............  INPUT INITIAL POLICY  .......................REM
1170 FOR I=1 TO Mstate
1175 READ #5;D(I)
1180 NEXT I
1185 GOTO 1350
1186 REM.............................................................REM
1190 READ Iprint
1195 READ Mstate
1200 READ Ndecision
1205 READ Maxi
1210 READ Maxtrial
1215 READ Discount
1220 REM...............  INPUT TRANSITION MATRIX  ....................REM
1225 FOR I=1 TO Mstate
1230 FOR K=1 TO Ndecision
1235 FOR J=1 TO Mstate
1240 READ P(I,J,K)
1245 NEXT J
1250 NEXT K
1255 NEXT I
```

```
1260 REM...............  INPUT COST/PROFIT MATRIX  ....................REM
1265 FOR I=1 TO Mstate
1270 FOR K=1 TO Ndecision
1275 REM    FOR J=1 TO Mstate
1280 REM    READ R(I,J,K)
1285 REM    NEXT J
1290 READ Q(I,K)
1295 NEXT K
1300 NEXT I
1305 REM...............  INPUT TIE-BREAKER FLAG  .....................REM
1310 FOR I=1 TO Mstate
1315 READ Tied(I)
1320 NEXT I
1325 REM...............  INPUT INITIAL POLICY  ........................REM
1330 FOR I=1 TO Mstate
1335 READ D(I)
1340 NEXT I
1345 REM...........................................................REM
1350 FOR I=1 TO Mstate
1355 FOR K=1 TO Ndecision
1360 IF Q(I,K)<>0 THEN 1425
1365 NEXT K
1370 NEXT I
1375 FOR I=1 TO Mstate
1380 FOR K=1 TO Ndecision
1385 Q(I,K)=0
1390 FOR J=1 TO Mstate
1395 Q(I,K)=Q(I,K)+P(I,J,K)*R(I,J,K)
1400 NEXT J
1405 NEXT K
1410 NEXT I
1415 REM...........................................................REM
1420 REM
1425 GOSUB 10205                 !  PRINT INPUT INFORMATION
1430 REM
1435 REM...........................................................REM
1440 Discount=1/(1+Discount/100)
2000 REM...........................................................REM
2032 IF Iprint>0 THEN GOSUB 10410
2034 Trial=0
2035 Trial=Trial+1
2036 IF Iprint>0 THEN GOSUB 10510
2050 REM..............  VALUE DETERMINATION  ........................REM
2060 FOR I=1 TO Mstate
2070 K=D(I)
2090 FOR J=1 TO Mstate
2100 A(I,J)=-P(I,J,K)*Discount
2110 IF I=J THEN A(I,J)=1+A(I,J)
2122 NEXT J
2124 IF Discount=1 THEN A(I,Mstate)=1
2125 A(I,Mstate+1)=Q(I,K)
2130 NEXT I
2140 N=Mstate
2141 Indic=1
2142 Eps=1E-20
```

```
2150 IF Iprint>1 THEN GOSUB 10610
2170 GOSUB 5000
2180 IF Discount<1 THEN 2300
2190 G=X(Mstate)
2220 X(Mstate)=0
2300 IF Iprint>0 THEN GOSUB 10710
3010 REM...............  POLICY IMPROVEMENT  ........................REM
3020 FOR I=1 TO Mstate
3025 FOR K=1 TO Ndecision
3026 Sum(I,K)=0
3030 FOR J=1 TO Mstate
3040 Sum(I,K)=Sum(I,K)+P(I,J,K)*X(J)
3041 NEXT J
3042 Sum(I,K)=Sum(I,K)*Discount+Q(I,K)
3054 IF Discount=1 THEN Sum(I,K)=Sum(I,K)-X(I)
3058 NEXT K
3060 E=Sum(I,1)
3070 Dd(I)=1
3106 FOR K=2 TO Ndecision
3110 IF Maxi>0 THEN 3150
3120 IF E<=Sum(I,K) THEN 3170
3130 E=Sum(I,K)
3140 Dd(I)=K
3142 GOTO 3200
3150 IF E>=Sum(I,K) THEN 3170
3160 GOTO 3130
3170 IF Sum(I,K)<>E THEN 3200
3190 IF Tied(I)>0 THEN Dd(I)=K
3200 NEXT K
3202 IF Iprint>0 THEN GOSUB 10810
3310 NEXT I
3320 FOR I=1 TO Mstate
3330 IF D(I)<>Dd(I) THEN 3360
3340 NEXT I
3350 GOTO 10905
3360 IF Iprint>0 THEN GOSUB 11010
3362 FOR I=1 TO Mstate
3370 D(I)=Dd(I)
3380 NEXT I
3390 IF Trial<Maxtrial THEN 2035
3410 GOTO 11110
5000 REM
5005 REM
5006 REM . . . . . FUNCTION SIMUL ( N,A,X,EPS,INDIC,NRC )
5007 REM
5008 REM   INDIC=-1, COMPUTE THE INVERSE OF THE N X N MATRIX
5009 REM   INDIC= 0, THE SET OF EQUATIONS A(N,N)*X(N)=A(N+1,N+1) IS
5010 REM             SOLVED AND THE INVERSE IS COMPUTED
5011 REM   INDIC=+1, THE SET OF EQUATIONS A(N,N)*X(N)=A(N+1,N+1) IS
5012 REM             SOLVED BUT THE INVERSE IS NOT COMPUTED
5013 REM
5014 REM   EPS  =MINIMUM ALLOWABLE VAULE FOR A PIVOT ELEMENT
5015 REM   A    =AUGMENTED MATRIX OF COEFFICIENT, A=A(I,J)
5016 REM
```

```
5017 REM   N     =NUMBER OF ROWS IN A
5018 REM
5019 REM   X     =SOLUTION VECTOR, X=X(I)
5020 REM
5021 Max=N
5022 IF Indic>=0 THEN Max=N+1
5023 REM
5024 REM . . . . . IS N LARGER THAN 50 . . . . .
5025 IF N<=50 THEN 5031
5026 PRINT #6;"      N IS GREATER THAN 50"
5027 Simul=0
5028 RETURN
5029 REM
5030 REM . . . . . BEGIN ELIMINATION PROCEDURE . . . . .
5031 Deter=1
5032 FOR K=1 TO N
5033 Km1=K-1
5034 REM
5035 REM . . . . . SEARCH FOR THE PIVOT ELEMENT . . . . .
5036 Pivot=0
5037 FOR I=1 TO N
5038 FOR J=1 TO N
5039 REM
5040 REM . . . . . SCAN IROW AND JCOL ARRARYS FOR INVALID PIVOT SUBSCRIPTS
5041 IF K=1 THEN 5048
5042 FOR Iscan=1 TO Km1
5043 FOR Jscan=1 TO Km1
5044 IF I=Irow(Iscan) THEN 5052
5045 IF J=Jcol(Jscan) THEN 5052
5046 NEXT Jscan
5047 NEXT Iscan
5048 IF ABS(A(I,J))<=ABS(Pivot) THEN 5052
5049 Pivot=A(I,J)
5050 Irow(K)=I
5051 Jcol(K)=J
5052 NEXT J
5053 NEXT I
5054 REM
5055 REM . . . . . INSURE THAT SELECTED PIVOT IS LARGER THAN EPS . . . . .
5056 IF ABS(Pivot)>Eps THEN 5062
5057 PRINT #6;"      ABS(PIVOT)=";ABS(Pivot);" IS LESS THEN ";Eps
5058 Simul=0
5059 RETURN
5060 REM
5061 REM . . . . . UPDATE THE DETERMINANT VALUE . . . . .
5062 Irowk=Irow(K)
5063 Jcolk=Jcol(K)
5064 Deter=Deter*Pivot
5065 REM
5066 REM . . . . . NORMALIZE PIVOT ROW ELEMENTS . . . . .
5067 FOR J=1 TO Max
5068 A(Irowk,J)=A(Irowk,J)/Pivot
5069 NEXT J
5070 REM
```

```
5071 REM . . . . . CARRY OUT ELIMINATION AND DEVELOP INVERSE . . . . .
5072 A(Irowk,Jcolk)=1/Pivot
5073 FOR I=1 TO N
5074 Aijck=A(I,Jcolk)
5075 IF I=Irowk THEN 5080
5076 A(I,Jcolk)=-Aijck/Pivot
5077 FOR J=1 TO Max
5078 IF J<>Jcolk THEN A(I,J)=A(I,J)-Aijck*A(Irowk,J)
5079 NEXT J
5080 NEXT I
5081 NEXT K
5082 REM
5083 REM . . . . . ORDER SOLUTION VALUES (IF ANY) AND CREAT JORD ARRAY
5084 FOR I=1 TO N
5085 Irowi=Irow(I)
5086 Jcoli=Jcol(I)
5087 Jord(Irowi)=Jcoli
5088 IF Indic>=0 THEN X(Jcoli)=A(Irowi,Max)
5089 NEXT I
5090 REM
5091 REM . . . . . ADJUST SIGN OF DETERMINANT
5092 Ich=0
5093 Nm1=N-1
5094 FOR I=1 TO Nm1
5095 Ip1=I+1
5096 FOR J=Ip1 TO N
5097 IF Jord(J)>=Jord(I) THEN 5102
5098 Jtemp=Jord(J)
5099 Jord(J)=Jord(I)
5100 Jord(I)=Temp
5101 Ich=Ich+1
5102 NEXT J
5103 NEXT I
5104 IF Ich/2*2<>Ich THEN Deter=-Deter
5105 Simul=Deter
5106 REM
5107 REM . . . . . END OF SUBROUTINE . . . . .
5108 RETURN
5109 REM
10100 REM.........................................................REM
10110 PRINT #Ko
10120 PRINT #Ko
10130 PRINT #Ko
10140 RETURN
10200 REM.........................................................REM
10205 COMMAND "M F H Hp-Ib#1"
10210 PRINT #Ko;LIN(2);TAB(10);"* * * * * INPUT INFORMATION * * * * *"
10220 PRINT #Ko;LIN(2);TAB(10);"MSTATE     ";Mstate
10225 PRINT #Ko;TAB(10);"NDECISION ";Ndecision
10230 PRINT #Ko;TAB(10);"MAXTRIAL  ";Maxtrial
10235 PRINT #Ko;TAB(10);"MAXIMUM   ";Maxi
10240 PRINT #Ko;TAB(10);"DISCOUNT  ";Discount
10250 PRINT #Ko;LIN(1);TAB(10);"STATE","      ";"DECISION","TIE-BREAKER"
10255 PRINT #Ko;TAB(10);"  I   ","      ";"  D(I)   ","  TIED(I)  "
```

```
10260 PRINT #Ko;TAB(10);"------","      ";"----------","--------------"
10265 FOR I=1 TO Mstate
10270 PRINT #Ko;TAB(10);I,"      ";D(I),Tied(I)
10275 NEXT I
10280 IF Iprint<2 THEN RETURN
10300 PRINT #Ko;LIN(1);TAB(10);"  STATE   "," DECISION ","  STATE   "
10302 PRINT #Ko;TAB(10);"  STATE   "," DECISION "," STATE "
10304 PRINT #Ko;TAB(10);"AT CURRENT","AT CURRENT"," AT NEXT   ","PROBABILITY"
10306 PRINT #Ko;TAB(10);"  STAGE   ","   STAGE   ","  STAGE    ","  P(I,J,K) "
10308 PRINT #Ko;TAB(10);"  ( I )   ","   ( K )   ","  ( J )    ","  P(I,J,K) "
10310 PRINT #Ko;TAB(10);"----------","----------","----------","-----------"
10312 FOR I=1 TO Mstate
10314 PRINT #Ko
10316 FOR K=1 TO Ndecision
10318 PRINT #Ko
10320 FOR J=1 TO Mstate
10322 PRINT #Ko;TAB(10);I,"",K,J,P(I,J,K)
10324 NEXT J
10326 NEXT K
10328 NEXT I
10330 PRINT #Ko;LIN(2);TAB(10);"  STATE   "," DECISION "
10332 PRINT #Ko;TAB(10);"AT CURRENT","AT CURRENT"
10334 PRINT #Ko;TAB(10);"  STAGE   ","   STAGE   ","COST/PROFIT"
10336 PRINT #Ko;TAB(10);"  ( I )   ","   ( K )   ","   Q(I,K)  "
10338 PRINT #Ko;TAB(10);"----------","----------","----------"
10340 FOR I=1 TO Mstate
10344 FOR K=1 TO Ndecision
10346 PRINT #Ko;TAB(10);I,"",K,Q(I,K)
10348 NEXT K
10350 PRINT #Ko
10352 NEXT I
10360 RETURN
10400 REM.............................................................REM
10410 PRINT #Ko;LIN(2);TAB(10);"  * * * * * INTERMEDIATE RESULTS * * * * *"
10420 RETURN
10500 REM.............................................................REM
10510 PRINT #Ko
10520 PRINT #Ko;" .........  TRIAL =";Trial
10530 RETURN
10600 REM.............................................................REM
10610 PRINT #Ko
10640 FOR I=1 TO Mstate
10650 FOR J=1 TO Mstate+1 STEP 4
10660 PRINT #Ko;TAB(10);"A(";I;",";J;")=";A(I,J);"A(";I;",";J+1;")=";A(I,J+1);
10665 PRINT #Ko;"A(";I;",";J+2;")=";A(I,J+2);"A(";I;",";J+3;")=";A(I,J+3)
10670 NEXT J
10675 PRINT #Ko
10680 NEXT I
10690 RETURN
10700 REM.............................................................REM
10710 PRINT #Ko
10720 IF Discount=1 THEN PRINT #Ko;TAB(10);"G      =";G
10730 FOR I=1 TO Mstate
10740 PRINT #Ko;TAB(10);"V(";I;")=";X(I)
10750 NEXT I
```

```
10760 PRINT #Ko
10770 RETURN
10800 REM.............................................................REM
10810 PRINT #Ko;LIN(1);TAB(10);"STATE","DECISION","VALUE"
10820 PRINT #Ko;TAB(10);"-----","--------","-----"
10830 FOR K=1 TO Ndecision
10840 PRINT #Ko;TAB(10);I,K,Sum(I,K)
10850 NEXT K
10860 RETURN
10900 REM.............................................................REM
10905 PRINT #Ko;LIN(3);
10910 PRINT #Ko;TAB(10);"* * * * * * *   FINAL RESULTS   * * * * * * * "
10915 PRINT #Ko;LIN(3)
10920 IF Discount=1 THEN PRINT #Ko;TAB(10);"LONG-TERM AVERAGE COST/RETURN=";G
10925 PRINT #Ko;LIN(2);TAB(10);"STATE","      ";"POLICY","VALUE"
10930 PRINT #Ko;TAB(10);"-----","      ";"------","-----"
10935 FOR I=1 TO Mstate
10940 K=D(I)
10945 PRINT #Ko;TAB(10);I,"      ";D(I),Sum(I,K)
10950 NEXT I
10955 GOTO 11210
11000 REM.............................................................REM
11010 PRINT #Ko;LIN(2)
11012 FOR L=1 TO Mstate
11014 PRINT #Ko;TAB(10);"OLD-D(";L;")=";D(L),"NEW-D(";L;")=";Dd(L)
11016 NEXT L
11020 RETURN
11100 REM.............................................................REM
11110 PRINT #Ko;LIN(3);TAB(20);"$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$"
11120 PRINT #Ko;TAB(20);"EXCEEDED THE MAXIMUM NUMBER OF TRIALS =";Maxtrial
11130 PRINT #Ko;TAB(20);"$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$"
11200 REM.............................................................REM
11210 PRINT #Ko;LIN(5);TAB(20);"* * * * * END OF TASK * * * * *"
11220 COMMAND "M f h hp-Ib#1"
60000 END
```